

实验报告

**大数据处理技术**

报 告 人 姓 名 刘飞鸿

学 号 S2010W0748

学 科 专 业 电子信息

报告提交日期 2020年11月3日

# 实验 5 NoSQL 和关系数据库的操作比较

## 一、实验目的

* 理解四种数据库(MySQL、HBase、Redis 和 MongoDB)的概念以及不同点；
* 熟练使用四种数据库操作常用的 Shell 命令；
* 熟悉四种数据库操作常用的 Java API。

## 二、实验平台

* + 操作系统：Linux（建议 Ubuntu16.04）；
  + Hadoop 版本：2.7.1；
  + MySQL 版本：5.6；
  + HBase 版本：1.1.2；
  + Redis 版本：3.0.6；
  + MongoDB 版本：3.2.6；
  + JDK 版本：1.7 或以上版本；
  + Java IDE：Eclipse；

## 三、实验步骤

#### （一） MySQL 数据库操作

学生表 Student

|  |  |  |  |
| --- | --- | --- | --- |
| Name | English | Math | Computer |
| zhangsan | 69 | 86 | 77 |
| lisi | 55 | 100 | 88 |

1. 根据上面给出的 Student 表，在 MySQL 数据库中完成如下操作：
2. 在 MySQL 中创建 Student 表，并录入数据；
3. 用 SQL 语句输出 Student 表中的所有记录；
4. 查询 zhangsan 的 Computer 成绩；
5. 修改 lisi 的 Math 成绩，改为 95。
6. 根据上面已经设计出的 Student 表，使用 MySQL 的 JAVA 客户端编程实现以下操作：
7. 向 Student 表中添加如下所示的一条记录：

|  |  |  |  |
| --- | --- | --- | --- |
| scofield | 45 | 89 | 100 |

1. 获取 scofield 的 English 成绩信息

#### （二）HBase 数据库操作

学生表 Student

|  |  |  |  |
| --- | --- | --- | --- |
| name | score | | |
| English | Math | Computer |
| zhangsan | 69 | 86 | 77 |
| lisi | 55 | 100 | 88 |

1. 根据上面给出的学生表 Student 的信息，执行如下操作：
2. 用 Hbase Shell 命令创建学生表 Student；
3. 用 scan 命令浏览 Student 表的相关信息；
4. 查询 zhangsan 的 Computer 成绩；
5. 修改 lisi 的 Math 成绩，改为 95。
6. 根据上面已经设计出的 Student 表，用 HBase API 编程实现以下操作：
7. 添加数据：English:45 Math:89 Computer:100

|  |  |  |  |
| --- | --- | --- | --- |
| scofield | 45 | 89 | 100 |

1. 获取 scofield 的 English 成绩信息。

#### （三）Redis 数据库操作

Student 键值对如下：

zhangsan:｛

English: 69

Math: 86

Computer: 77

｝

lisi:｛

English: 55

Math: 100

Computer: 88

｝

1. 根据上面给出的键值对，完成如下操作：
   1. 用 Redis 的哈希结构设计出学生表 Student（键值可以用 student.zhangsan 和

student.lisi 来表示两个键值属于同一个表）；

* 1. 用 hgetall 命令分别输出 zhangsan 和 lisi 的成绩信息；
  2. 用 hget 命令查询 zhangsan 的 Computer 成绩；
  3. 修改 lisi 的 Math 成绩，改为 95。

1. 根据上面已经设计出的学生表Student，用Redis 的JAVA 客户端编程(jedis)，实现如下操作：
2. 添加数据：English:45 Math:89 Computer:100

该数据对应的键值对形式如下：

scofield:｛

English: 45

Math: 89

Computer: 100

｝

1. 获取 scofield 的 English 成绩信息

#### （四）MongoDB 数据库操作

Student 文档如下:

{

“name”: “zhangsan”, “score”: {

“English”: 69,

“Math”: 86,

“Computer”: 77

}

}

{

“name”: “lisi”, “score”: {

“English”: 55,

“Math”: 100,

“Computer”: 88

}

}

1. 根据上面给出的文档，完成如下操作：
   1. 用 MongoDB Shell 设计出 student 集合；
   2. 用 find()方法输出两个学生的信息；
   3. 用 find()方法查询 zhangsan 的所有成绩(只显示 score 列)；
   4. 修改 lisi 的 Math 成绩，改为 95。
2. 根据上面已经设计出的 Student 集合，用 MongoDB 的 Java 客户端编程，实现如下操作：
3. 添加数据：English:45 Math:89 Computer:100

与上述数据对应的文档形式如下：

{

“name”: “scofield”, “score”: {

“English”: 45,

“Math”: 89,

“Computer”: 100

}

}

1. 获取 scofield 的所有成绩成绩信息(只显示 score 列)

## 四、实验报告

* **HBase搭建环境、安装部署**

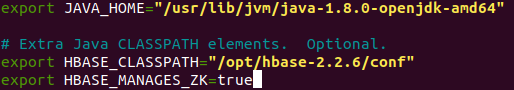
1. 下载HBase发布包，通过上传软件上传至/usr/local/src目录
2. 解压软件包并移动至/opt/目录

|  |
| --- |
| cd /usr/local/src  tar -xzvf hbase-2.2.6-bin.tar.gz  mv ./hbase-2.2.6 /opt/hbase-2.2.6 |

1. 配置环境变量

打开etc/profile:



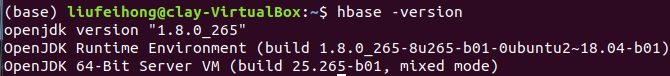
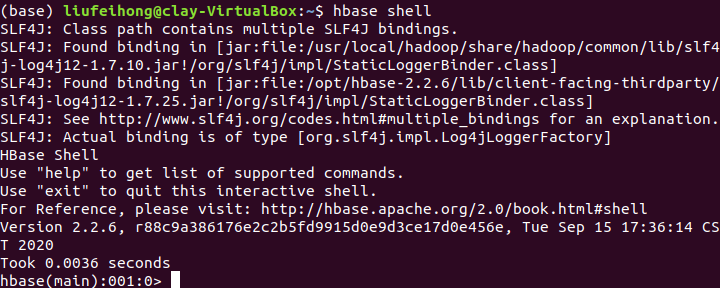
1. 配置相关的文件
   * + 1. 配置hbase-env.sh 
       2. 配置hbase-site.xml

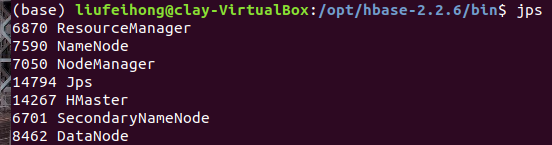
|  |
| --- |
| <property>  <name>hbase.rootdir</name>  <value>hdfs://localhost:9000/hbase</value>  </property>  <property>  <name>hbase.cluster.distributed</name>  <value>true</value>  </property> |

* + - 1. 拷贝Hadoop的hdfs-site.xml文件至${HBASE\_HOME}/conf 目录

1. 添加HBase权限:

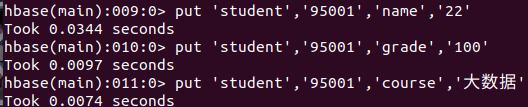
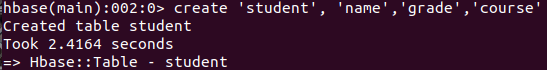


1. 验证是否安装成功:
   * + 1. 输入 hbase -version: 
       2. 输入 hbase shell
       3. 输入jps看到

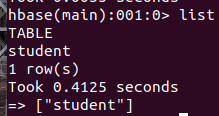


* **利用编程实现以下指定功能，并用 Hadoop 提供的 HBase Shell 命令完成相同任务：**

1. 列出 HBase 所有的表的相关信息，例如表名；
2. 创建一个表并添加数据:

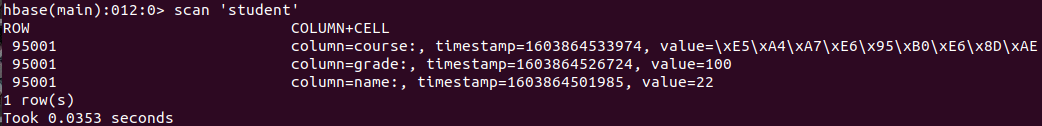


1. 查看所有表:



1. 在终端打印出指定的表的所有记录数据；

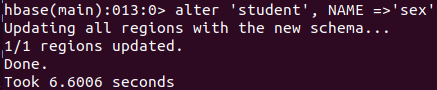
将Student的所有记录打印出来



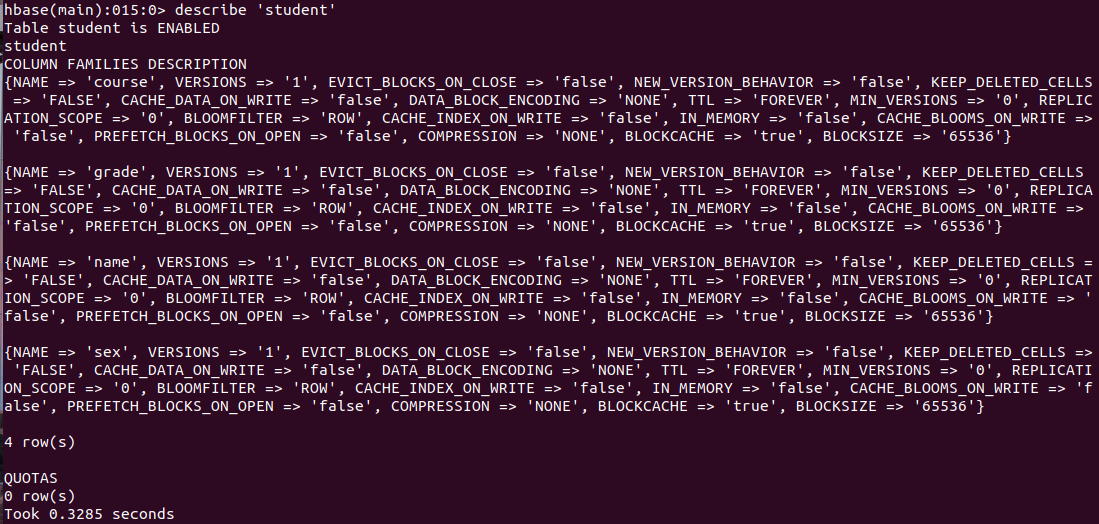
可以将course十六进制对应的中文打印出来：



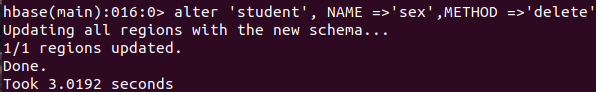
1. 向已经创建好的表添加和删除指定的列族或列；
   * 添加列sex:



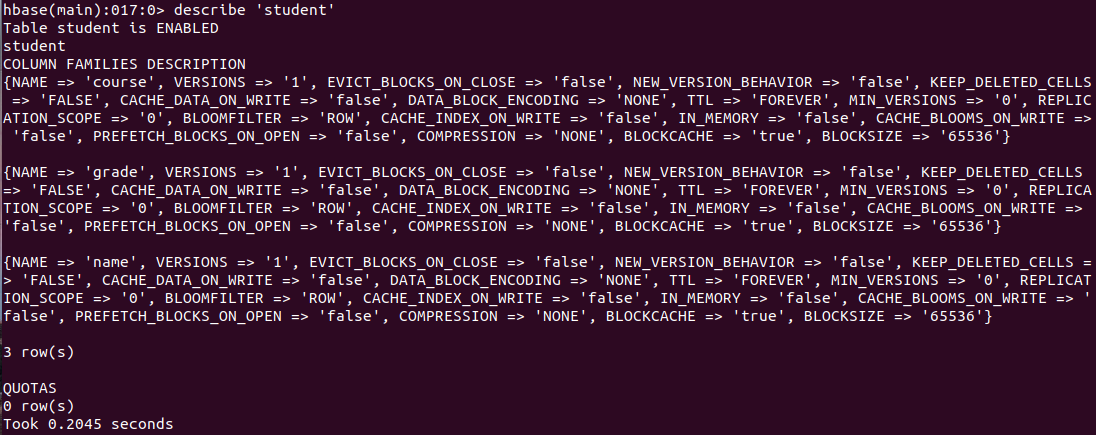
验证是否添加成功:



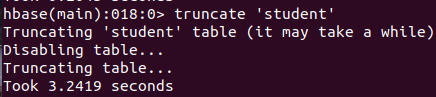
* 删除列sex:



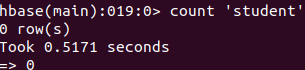
验证是否删除成功:



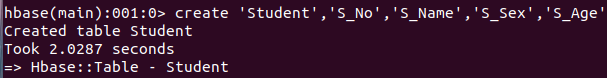
1. 清空指定的表的所有记录数据；



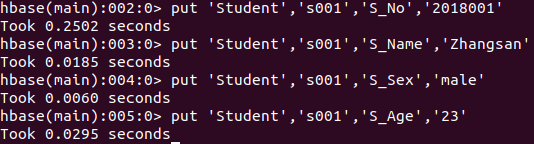
1. 统计表的行数。



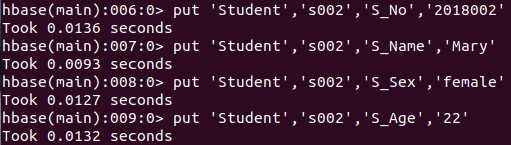
* **现有以下关系型数据库中的表和数据，要求将其转换为适合于 HBase 存储的表并插入数据：**

1. **学生表创建及插入: **

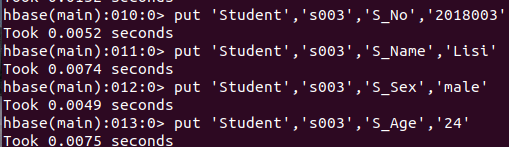
插入第一行数据:



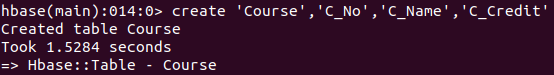
插入第二行数据:



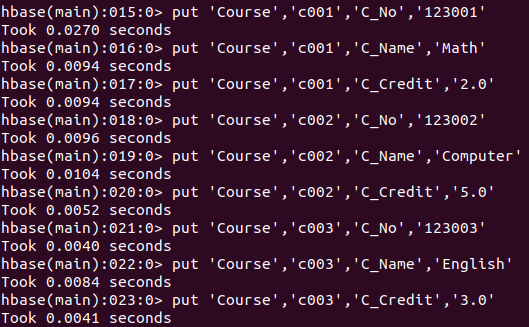
插入第三行数据:



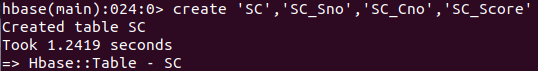
1. **课程表创建及插入:**

****

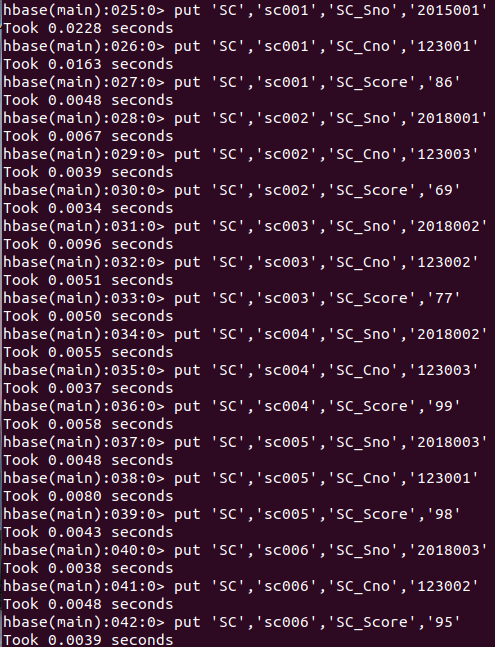
插入数据:



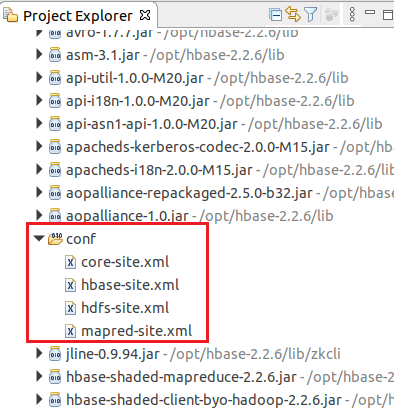
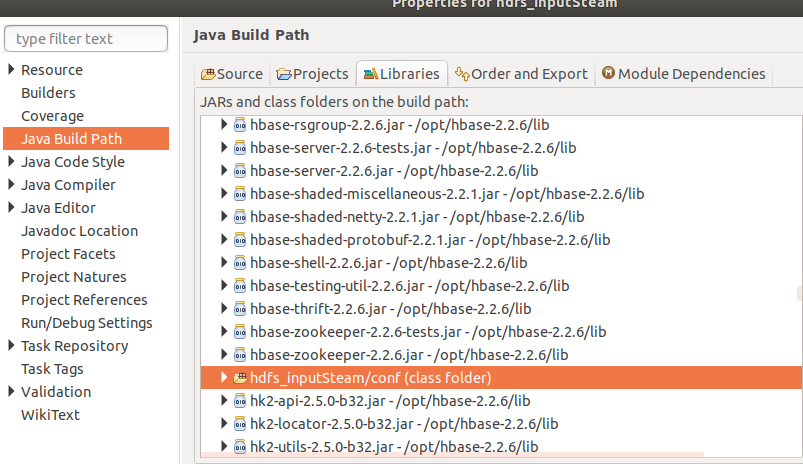
1. **选课表创建及插入:**

****

插入数据:



* 请编程实现以下功能：
  + - 1. createTable(String tableName, String[] fields)创建表，参数 tableName 为表的名称，字符串数组 fields 为存储记录各个字段名称的数组。要求当 HBase 已经存在名为 tableName 的表的时候，先删除原有的表，然后再创建新的表。
         1. 在eclipse下配置hbase开发环境:
    - 在工程中添加所需的jar包: 我们需要的jar包在hbase的安装(解压缩)目录下的lib目录中
    - 指定HBase配置文件的位置: 将HBase的配置文件复制一份到工程里。先在工程目录下创建一个名为conf的目录，再将HBase的配置文件 hbase-site.xml以及hadoop的配置文件core-site.xml、hdfs-site.xml、mapred-site.xml三个文件复制到该目录下。接着，还是右击项目工程，选择 Properties->Java Build Path->Libraries->Add Class Folder, 将刚刚增加的conf目录选上：



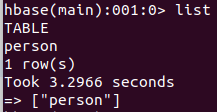
* + - * 1. 运行的代码如下：

|  |
| --- |
| import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.hbase.HBaseConfiguration;  import org.apache.hadoop.hbase.HColumnDescriptor;  import org.apache.hadoop.hbase.HTableDescriptor;  import org.apache.hadoop.hbase.TableName;  import org.apache.hadoop.hbase.client.Admin;  import org.apache.hadoop.hbase.client.Connection;  import org.apache.hadoop.hbase.client.ConnectionFactory;  import java.io.IOException;  public class CreateTable {  public static Configuration configuration;  public static Connection connection;  public static Admin admin;  public static void createTable(String tableName, String[] fields) throws IOException {  init();  TableName tablename = TableName.valueOf(tableName);  if (admin.tableExists(tablename)) {  System.out.println("table is exists!");  admin.disableTable(tablename);  admin.deleteTable(tablename);  }  HTableDescriptor hTableDescriptor = new HTableDescriptor(tablename);  for (String str : fields) {  HColumnDescriptor hColumnDescriptor = new HColumnDescriptor(str);  hTableDescriptor.addFamily(hColumnDescriptor);  }  admin.createTable(hTableDescriptor);  close();  }  public static void init() {  configuration = HBaseConfiguration.create();  configuration.set("hbase.rootdir", "hdfs://localhost:9000/hbase");  try {  connection = ConnectionFactory.createConnection(configuration);  admin = connection.getAdmin();  } catch (IOException e) {  e.printStackTrace();  }  }  public static void close() {  try {  if (admin != null) {  admin.close();  }  if (null != connection) {  connection.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  String[] fields = {"Score"};  try {  createTable("person", fields);  } catch (IOException e) {  e.printStackTrace();  }  }  } |

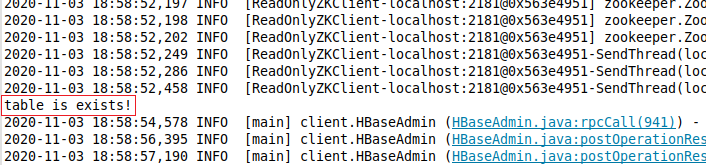
* + - * 1. 运行的结果如下：



* + - * 1. 通过shell命令验证：

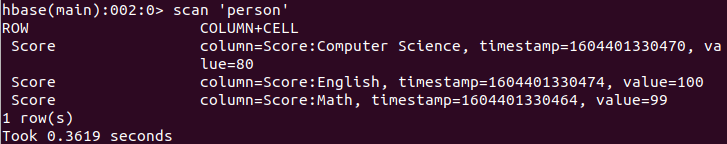


* + - * 1. 如果再次运行代码, 输出“table is exists!”:



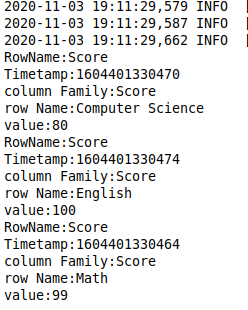
* + - 1. addRecord(String tableName, String row, String[] fields, String[] values)向表 tableName、行 row（用 S\_Name 表示）和字符串数组 fields 指定的单元格中添加对应的数据 values。其中，fields 中每个元素如果对应的列族下还有相应的列限定符的话，用“columnFamily:column”表示。例如，同时向“Math”、“Computer Science”、“English”三列添加成绩时，字符串数组 fields 为{“Score:Math”, ”Score:Computer Science”, ”Score:English”}， 数组 values 存储这三门课的成绩。
         1. 运行的代码如下：

|  |
| --- |
| package hdfs\_inputSteam;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.hbase.HBaseConfiguration;  import org.apache.hadoop.hbase.TableName;  import org.apache.hadoop.hbase.client.\*;  import java.io.IOException;  public class AddRecord {  public static Configuration configuration;  public static Connection connection;  public static Admin admin;  public static void addRecord(String tableName, String row, String[] fields, String[] values) throws IOException {  init();  Table table = connection.getTable(TableName.valueOf(tableName));  for (int i = 0; i != fields.length; i++) {  Put put = new Put(row.getBytes());  String[] cols = fields[i].split(":");  put.addColumn(cols[0].getBytes(), cols[1].getBytes(), values[i].getBytes());  table.put(put);  }  table.close();  close();  }  public static void init() {  configuration = HBaseConfiguration.create();  configuration.set("hbase.rootdir", "hdfs://localhost:9000/hbase");  try {  connection = ConnectionFactory.createConnection(configuration);  admin = connection.getAdmin();  } catch (IOException e) {  e.printStackTrace();  }  }  public static void close() {  try {  if (admin != null) {  admin.close();  }  if (null != connection) {  connection.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  String[] fields = {"Score:Math", "Score:Computer Science", "Score:English"};  String[] values = {"99", "80", "100"};  try {  addRecord("person", "Score", fields, values);  } catch (IOException e) {  e.printStackTrace();  }  }  } |

* + - * 1. 对程序进行验证：  
           
      1. scanColumn(String tableName, String column)浏览表 tableName 某一列的数据，如果某一行记录中该列数据不存在，则返回 null。要求当参数 column 为某一列族名称时，如果底下有若干个列限定符，则要列出每个列限定符代表的列的数据；当参数 column 为某一列具体名称（例如“Score:Math”）时，只需要列出该列的数据。
         1. 运行的代码如下：

|  |
| --- |
| package hdfs\_inputSteam;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.hbase.Cell;  import org.apache.hadoop.hbase.CellUtil;  import org.apache.hadoop.hbase.HBaseConfiguration;  import org.apache.hadoop.hbase.TableName;  import org.apache.hadoop.hbase.client.\*;  import org.apache.hadoop.hbase.util.Bytes;  import java.io.IOException;  public class ScanColumn {  public static Configuration configuration;  public static Connection connection;  public static Admin admin;  public static void scanColumn(String tableName, String column) throws IOException {  init();  Table table = connection.getTable(TableName.valueOf(tableName));  Scan scan = new Scan();  scan.addFamily(Bytes.toBytes(column));  ResultScanner scanner = table.getScanner(scan);  for (Result result = scanner.next(); result != null; result = scanner.next()) {  showCell(result);  }  table.close();  close();  }  public static void showCell(Result result) {  Cell[] cells = result.rawCells();  for (Cell cell : cells) {  System.out.println("RowName:" + new String(CellUtil.cloneRow(cell)) + " ");  System.out.println("Timetamp:" + cell.getTimestamp() + " ");  System.out.println("column Family:" + new String(CellUtil.cloneFamily(cell)) + " ");  System.out.println("row Name:" + new String(CellUtil.cloneQualifier(cell)) + " ");  System.out.println("value:" + new String(CellUtil.cloneValue(cell)) + " ");  }  }  public static void init() {  configuration = HBaseConfiguration.create();  configuration.set("hbase.rootdir", "hdfs://localhost:9000/hbase");  try {  connection = ConnectionFactory.createConnection(configuration);  admin = connection.getAdmin();  } catch (IOException e) {  e.printStackTrace();  }  }  // 关闭连接  public static void close() {  try {  if (admin != null) {  admin.close();  }  if (null != connection) {  connection.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  try {  scanColumn("person", "Score");  } catch (IOException e) {  e.printStackTrace();  }  }  } |

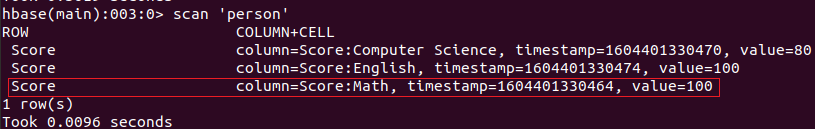
* + - * 1. 运行的结果如下：



* + - 1. modifyData(String tableName, String row, String column)修改表 tableName，行 row（可以用学生姓名 S\_Name 表示），列 column 指定的单元格的数据。
         1. 代码如下：

|  |
| --- |
| package hdfs\_inputSteam;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.hbase.Cell;  import org.apache.hadoop.hbase.HBaseConfiguration;  import org.apache.hadoop.hbase.TableName;  import org.apache.hadoop.hbase.client.\*;  import java.io.IOException;  public class ModifyData {  public static long ts;  public static Configuration configuration;  public static Connection connection;  public static Admin admin;  public static void modifyData(String tableName, String row, String column, String val) throws IOException {  init();  Table table = connection.getTable(TableName.valueOf(tableName));  Put put = new Put(row.getBytes());  Scan scan = new Scan();  ResultScanner resultScanner = table.getScanner(scan);  for (Result r : resultScanner) {  for (Cell cell : r.getColumnCells(row.getBytes(), column.getBytes())) {  ts = cell.getTimestamp();  }  }  put.addColumn(row.getBytes(), column.getBytes(), ts, val.getBytes());  table.put(put);  table.close();  close();  }  public static void init() {  configuration = HBaseConfiguration.create();  configuration.set("hbase.rootdir", "hdfs://localhost:9000/hbase");  try {  connection = ConnectionFactory.createConnection(configuration);  admin = connection.getAdmin();  } catch (IOException e) {  e.printStackTrace();  }  }  public static void close() {  try {  if (admin != null) {  admin.close();  }  if (null != connection) {  connection.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  try {  modifyData("person", "Score", "Math", "100");  } catch (IOException e) {  e.printStackTrace();  }  }  } |

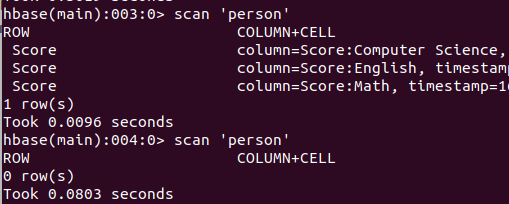
* + - * 1. 验证如下：



* + - 1. deleteRow(String tableName, String row)删除表 tableName 中 row 指定的行的记录。
         1. 运行的代码如下：

|  |
| --- |
| package hdfs\_inputSteam;  import org.apache.hadoop.conf.Configuration;  import org.apache.hadoop.hbase.Cell;  import org.apache.hadoop.hbase.HBaseConfiguration;  import org.apache.hadoop.hbase.TableName;  import org.apache.hadoop.hbase.client.\*;  import org.apache.hadoop.hbase.util.Bytes;  import java.io.IOException;  public class DeleteRow {  public static long ts;  public static Configuration configuration;  public static Connection connection;  public static Admin admin;  public static void deleteRow(String tableName, String row) throws IOException {  init();  Table table = connection.getTable(TableName.valueOf(tableName));  Delete delete=new Delete(row.getBytes());  table.delete(delete);  table.close();  close();  }  public static void init() {  configuration = HBaseConfiguration.create();  configuration.set("hbase.rootdir", "hdfs://localhost:9000/hbase");  try {  connection = ConnectionFactory.createConnection(configuration);  admin = connection.getAdmin();  } catch (IOException e) {  e.printStackTrace();  }  }  public static void close() {  try {  if (admin != null) {  admin.close();  }  if (null != connection) {  connection.close();  }  } catch (IOException e) {  e.printStackTrace();  }  }  public static void main(String[] args) {  try {  deleteRow("person", "Score");  } catch (IOException e) {  e.printStackTrace();  }  }  } |

* + - * 1. 验证如下：



对比运行前和运行后的结果，可以看出删除一行数据成功！